CS10720 Problems and Solutions

Thomas Jansen

Today: Page Rank

April 18th

Plans for Today

1. Introduction and Motivation
   - History of Search Engines
   - History of Google
2. Web Search: Components
   - Matching
   - Ranking
3. Page Rank
   - Algorithm
4. Summary
   - Summary & Take Home Message

A Brief History of the Internet

1961 first packet-switching networks

1969 advanced research projects agency network (ARPANET) initiated by the United States Department of Defense

1970 Mark I network (first UK-based network; main figure Donald Davies (1924–2000) from Wales)

1976 X.25 transport protocol for packet-switching networks

1980 USENET based on UUCP

1982 TCP/IP protocol suite, formally introducing the Internet

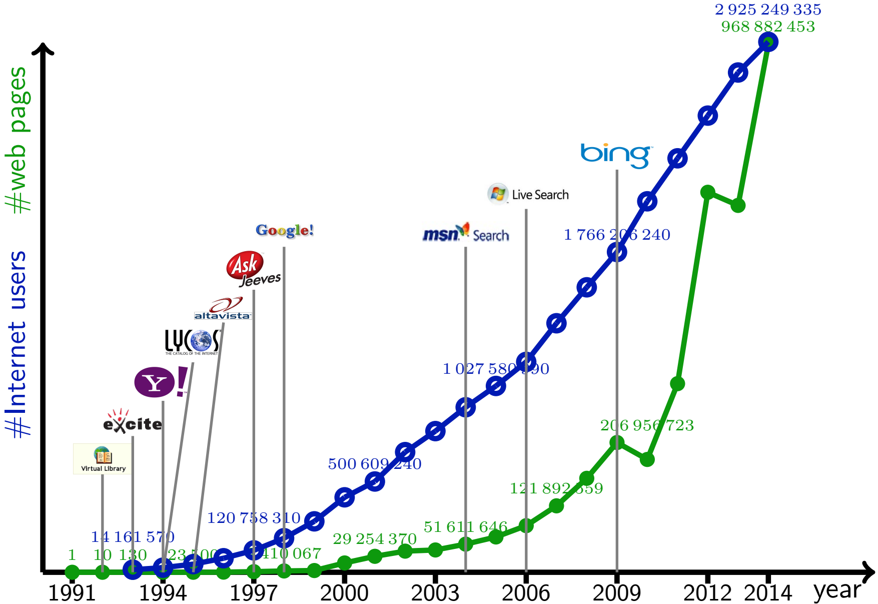
1982 SMTP (simple mail transfer protocol)

1983 DNS (domain name service)

1991 Gopher (application layer protocol, menu-based approach for document distribution)

1991 WWW (world wide web) and HTTP (hypertext transfer protocol)

WWW and Internet in Numbers



A Brief History of Google

1995 Sergej Brin and Larry Page meet at Stanford

1996 Brin and Page collaborate on research about web search, ranking hypertext and dynamic data mining

1996 Larry Page sets up BackRub, a web crawler at Stanford

1998 Brin and Page publish ‘The anatomy of a large-scale hypertextual Web search engine’ at *WWW7: Proceedings of the Seventh International Conference on World Wide Web* (and in the *Journal of Computer Networks and ISDN Systems* 30:107–117 (`http://doi.org/10.1016/S0169-7552(98)00110-X`)) introducing the name Google for the web search, with an architecture aiming at 100 000 000 web pages

08/1998 Andy Bechtolsheim writes $100 000 cheque for Google, Inc.

09/1998 Google, Inc. is registered

1999 Brin and Page try to sell Google to Excite for $1 000 000

1999 Brin and Page try to sell Google to Excite for $750 000

Web Search Components: Matching and Ranking

What is an input for a web search?

Possible inputs

- word (e. g., Google)
- several words (e. g., Google search)
- phrase (e. g., “search algorithm”)
- words and/or phrase with additional qualifiers (e. g., “exam timetable" site:aber.ac.uk)
- . . .

What do we expect from the results?

- list of matching web pages
- sorted according to relevance
- delivered promptly

Matching

Input word(s) and/or phrase, possibly with additional qualifiers

Problem find ‘all’ matching web pages fast

Observation fast implies

- cannot access web pages for search
- locally (with the search engine) store index information required
- web crawling should deliver index that supports different kinds of searches efficiently

Remark all matching web pages not realistic
(even when restricted to web pages in the search index)
$\Rightarrow$ ranking with respect to relevance needs to be incorporated

Still useful to conceptually consider matching and ranking separately

Here only ranking

Ranking Web Pages w. r. t Relevance to Search Terms

WHEN A USER TAKES A PHOTO, THE APP SHOULD CHECK WHETHER THEY'RE IN A NATIONAL PARK...
SURE, EASY GIS LOOKUP. GIMME A FEW HOURS.
... AND CHECK WHETHER THE PHOTO IS OF A BIRD.
I'LL NEED A RESEARCH TEAM AND FIVE YEARS.
IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.

https://xkcd.com/1425/

How can computers determine relevance of a web page w. r. t. search terms?

Fact That's really hard.

Idea try estimating relevance by exploiting existing structural information that is provided by people

Because in general, people really good at determining relevance

How can we do that?

Hope that existing links between web pages express relevance

Ideas to Exploit Structural Information

Remember we want to exploit existing structural information to estimate relevance of a web page

What kind of structural information is available in the WWW?

Observation hyperlinks are available (ignoring the semantic web; not available 1998)

Idea ❶ a page is more relevant if more pages link to it
i. e., rate relevance by counting incoming links

Are all pages of equal importance?

Idea ❷ a page is more relevant if more relevant pages link to it
i. e., rate relevance by adding relevance for incoming links
(relevance of link = relevance of originating web page)

On Surfing the Web

How does surfing the web work?

Typical (?) example on an arbitrary Saturday night
you are bored and decide to go to
`www.google.co.uk` to search for prime numbers
leading you to `http://en.wikipedia.org/wiki/Prime_number`
leading you to `http://en.wikipedia.org/wiki/Prime_number_theorem`
leading you to `http://en.wikipedia.org/wiki/Isabelle_(proof_assistant)`
leading you to `https://isabelle.in.tum.de/community/Projects`
leading you to `http://formare.github.io/auctions/`
when you get bored again and start over at `https://www.youtube.com` . . .

What is typical about this?

Observation 'surfing the web' usually means

- starting somewhere
- following a couple of links, sequentially
- stopping to follow and start somewhere else again

Idea for Page Rank

Random Surfer

1. Start on a random page.
2. Repeat forever
3. Either (with probability r) go to another random page or (with probability $1 - r$) follow one link (chosen uniformly at random) to another page.

Observation captures kind of web surfing described on previous slide to some degree
(Note replaces purposeful surfing with random decisions)

Disadvantage not very accurate
Advantages simple, manageable

Where is the idea for ranking?

Idea rank web pages in order that corresponds to probability that 'random surfer' is on them

Towards Page Rank

Random Surfer

1. Start on a random page.
2. Repeat forever
3. Either (with probability r) go to another random page or (with probability $1 - r$) follow one link (chosen uniformly at random) to another page.

Given a ‘web graph’ (i. e., web pages connected by links)
How can we find out the probability for each page that the ‘random surfer’ is on it?

Fact that’s hard
(at least in general, without restrictions on the graph)

Idea simulate the ‘random surfer’
and hope that the true probabilities are approached quickly

Remark approximations are acceptable because ‘random surfer’ model is only a crude approximation itself

Page Rank

Algorithm to compute Page Rank values (with error $< \varepsilon$) by approximating stationary probabilities for 'random surfer'

Notation

- set of all web pages: V
- number of all web pages: $n = |V|$
- number of different links from v somewhere: $L(v)$
- set of pages with links to v: $I(v)$
- probability of 'restart': r ($1 - r$ called damping factor)
- current estimate of the Page Rank of web page $v \in V$: $\mathsf{PR}(v)$

```
For all $v \in V$ set $\mathsf{PR}(v) := 1/n$.
Do
    Set $\Delta := 0$.
    For each $v \in V$ do
        $\mathsf{PR}_{\mathsf{new}}(v) := \frac{r}{n} + (1-r) \cdot \sum_{w \in I(v)} \frac{\mathsf{PR}(w)}{L(w)}$
        if $|\mathsf{PR}_{\mathsf{new}}(v) - \mathsf{PR}(v)| > \Delta$ then $\Delta := |\mathsf{PR}_{\mathsf{new}}(v) - \mathsf{PR}(v)|$
    For each $v \in V$ do
        $\mathsf{PR}(v) := \mathsf{PR}_{\mathsf{new}}(v)$
Until $\Delta < \epsilon$
```

Page Rank (in English)

1. Set Page Rank value for all pages to $1/(\text{number of pages})$ initially.

2. Work in rounds in the following way:

4.-5. Compute the new Page Rank value for v as $r/(\text{number of pages})$ plus, for each page with a link to v, $(1-r)$ times that page's Page Rank value divided by the number of different links leaving it.

6. Keep track of the greatest change in Page Rank values.

9. Stop when this difference decreases below ε.

```
1.  For all $v \in V$ set $\mathrm{PR}(v) := 1/n$.
2.  Do
3.      Set $\Delta := 0$.
4.      For each $v \in V$ do
5.          $\mathrm{PR}_{\text{new}}(v) := \frac{r}{n} + (1-r) \cdot \sum_{w \in L(v)} \frac{\mathrm{PR}(w)}{L(w)}$
6.          if $|\mathrm{PR}_{\text{new}}(v) - \mathrm{PR}(v)| > \Delta$ then $\Delta := |\mathrm{PR}_{\text{new}}(v) - \mathrm{PR}(v)|$
7.      For each $v \in V$ do
8.          $\mathrm{PR}(v) := \mathrm{PR}_{\text{new}}(v)$
9.  Until $\Delta < \epsilon$
```

Summary & Take Home Message

Things to remember

- history of the Internet
- history of web search
- history of Google
- web search: matching and ranking
- ranking ideas: popularity (i. e., number of links) and importance (i. e., rank of linking pages)
- idea: random surfer
- Page Rank algorithm

Take Home Message

- Page Rank is a relatively simple and extremely powerful and versatile ranking algorithm for graphs.
- Simple ideas can help earn lots of money but it's hard to recognise a good idea before it happens.

Lecture feedback `http://onlineted.com`